The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed M Parkin

Dated 26 July 2001

CERTIFIED COPY OF PRIORITY DOCUMENT

0300

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of

John Aram SAFA

Art Unit:

Application No: 09/905,571

Examiner:

Filed: July 13, 2001

For: DIGITAL DATA PROTECTION ARRANGEMENT

TRANSMITTAL OF CERTIFIED COPIES

Commissioner for Patents
Washington, D.C. 20231

Sir:

This application claims priority of United Kingdom Patent Application No. 0017481.3 filed July 18, 2000 and United Kingdom Patent Application No. 0102982.6 filed February 7, 2001. Certified copies of the United Kingdom patent applications are transmitted herewith in order to complete the claim for priority.

Respectfully submitted,

John Smith-Hill
Reg. No. 27,730

SMITH-HILL & BEDELL, P.C.
12670 N.W. Barnes Road, Suite 104
Portland, Oregon 97229

(503) 574-3100

Docket: SWIN 2277

Certificate of Mailing

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to the Commissioner for Patents, Washington, D.C. 20231, on the 10th day of September, 2001.

Patents Form 1/77

Patents Act 1977
(Rule 16)

The Patent Office

1/77

THE PATENT OFFICE
L
- 7 FEB 2001
NEWPORT

07FEB01 E603969-1 D02833
P01/7700 0.00-0102982.6

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office
Cardiff Road
Newport
Gwent NP9 1RH

1. Your reference MPS/7743

2. Patent application number
(The Patent Office will fill in this part) 0102982.6 07 FEB 2001

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*
BitArts Limited
Vernon House, 18 Friar Lane,
Nottingham, NG1 6DQ.

Patents ADP number *(if you know it)* 8077230 2 01

If the applicant is a corporate body, give the country/state of its incorporation United Kingdom

4. Title of the invention
Software Protection

5. Name of your agent *(if you have one)*
Swindell & Pearson

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*
48 Friar Gate,
Derby DE1 1GY

Patents ADP number *(if you know it)* 00001578001

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*
a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))
YES

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	
Description	11
Claim(s)	
Abstract	
Drawing(s)	3 + 3

10. If you are also filing any of the following, state how many against each item.

Priority documents	
Translations of priority documents	
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	
Request for preliminary examination and search *(Patents Form 9/77)*	
Request for substantive examination *(Patents Form 10/77)*	
Any other documents *(please specify)*	

11. I/We request the grant of a patent on the basis of this application.

Signature Swindell & Pearson Date 06/02/2001

Swindell & Pearson

12. Name and daytime telephone number of person to contact in the United Kingdom Mr. M.P. Skinner (01332) 367051

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) If you need help to fill in this form or you have any questions, please contact the Patent Office on 0645 500505.

b) Write your answers in capital letters using black ink or you may type them.

c) If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.

d) If you have answered 'Yes' Patents Form 7/77 will need to be filed.

e) Once you have filled in the form you must remember to sign and date it.

f) For details of the fee and ways to pay please contact the Patent Office.

SOFTWARE PROTECTION

The present invention relates to software protection and in particular, but not exclusively, to the protection of commercial software applications against unauthorised copying, use, modification or the like.

Many proposals have been made for protecting software. However, it has been found that even sophisticated proposals can be vulnerable to being undermined by a committed hacker. A hacker will often make use of debugger software in order to attack the security of software. Debugger programs are primarily intended for the legitimate operation of checking for faults in complex software. Debugger programs generally operate by inserting an interrupt command at a point in the software which the operator wishes to review. When execution of the software reaches the interrupt command, control is handed to the debugger program, which then looks at the next block of code, presenting the results to the operator in high or low level language, allowing the operator to check that block before allowing execution of the main program to continue.

One approach used by a hacker seeking to circumvent security routines is to use a debugger program to look for points within the main program at which security procedures are invoked. Jump commands are inserted at that point in order to jump over those routines when the main program is executed. Much modern software is sufficiently complex that the task of locating these points within the main program is arduous, but nevertheless, a skilled and determined hacker is likely to be able to find them eventually. Once found, the hacker can disable the security routines, which renders even the most sophisticated routine ineffective.

According to the present invention, there is provided a software protection arrangement which comprises first storage means containing protected software; and security means operable to authorise execution of the protected software in response to successful completion of one or more security

checks, the security means comprising:-

(a) at least one block of executable code which is stored in non-executable form and which requires execution to authorise execution of the protected software;

(b) conversion means operable to convert the said block of code to an executable form by means of an algorithm which requires at least one conversion key, and further operable to derive a conversion key, for use in the algorithm, by reference to a target block of code in executable or non-executable form;

whereby an appropriate conversion key will be derived only if the target block is unmodified.

The security means may comprise a plurality of blocks of executable code stored in non-executable form and each of which requires execution to authorise execution of the protected software, the conversion means being operable to convert each block to executable form. Conversion of each block is preferably achieved by a conversion key derived from a respective target block. At least one block is preferably operable, upon execution, to convert another block to an executable form for subsequent execution. Each block is preferably operable, upon execution, to convert another block to an executable form for subsequent execution.

The or each target block may be contained within the protected software or within the first security means.

The first storage means preferably stores the protected software in non-executable form, the security means being operable to convert the protected software to executable form. The protected software is preferably held in the first storage means in encrypted or compressed form.

Preferably the or each algorithm for converting code is a CRC algorithm. Preferably the or each block of code is initially in compressed and/or encrypted form.

The invention also provides computer software operable to provide protection for a second item of computer software, the protection software comprising security means operable to authorise execution of the protected software in response to successful completion of one or more security checks, and having at least one block of executable code which is stored in non-executable form and which requires execution to authorise execution of the protected software, and the protection software further comprising conversion means operable to convert the said block of code to an executable form by means of an algorithm which requires at least one conversion key, the conversion means being further operable to derive a conversion key, for use in the algorithm, by reference to a target block of code in executable or non-executable form, whereby an appropriate conversion key will be derived only if the target block is unmodified.

The invention also provides a computer memory device containing computer software as set out above.

The invention also provides a computer system containing an item of computer software protected by means of computer software as set out above.

Examples of the present invention will now be described in detail, by way of example only, and with reference to the accompanying drawings, in which:-

Fig. 1 is a schematic diagram of a computer system with which a software protection arrangement according to the invention may be implemented;

Fig. 2 is a schematic diagram of the contents of RAM of the system of Fig. 1, during implementation of the invention;

Fig. 3 corresponds with Fig. 2, showing an alternative form of implementation of the invention;

Fig. 4 schematically illustrates part of the listing of the security software illustrated in Figs. 2 and 3;

Fig. 5 schematically illustrates the end of the listing of the security software of Figs. 1 and 2; and

Fig. 6 schematically illustrates a further form of implementation of the invention.

Fig. 1 illustrates the basic components of a data processing system with which the invention can be implemented. A computer system 10 contains a processor 12 to which input/output devices 14 are connected. The processor 12 is provided with random access memory (RAM) 16 for use during processing. Additional memory capacity is provided at 18, for instance in the form of a hard drive.

It is common practice for a software application to be stored on the drive 18 until needed, and then to be loaded by compiling into a machine code version which is installed on the RAM 16, for speed of access and processing by the processor 12. The present invention relates to the treatment of the software after loading onto the RAM 16 and accordingly, no further description will be given of the operation of loading from memory 18 to RAM 16, these operations being conventional in themselves, and readily understood by the skilled reader.

Example 1

Fig. 2 illustrates a simple example of implementation of the invention. Alternative forms, which are more elaborate, are described below (Examples 2 and 3).

In Fig. 2, the RAM 16 is shown to contain two blocks of software 20,22. The first block 20 is the software protected by the arrangement of the invention. The block 22 is a security block which implements the protection arrangements. The protected software 20 includes an application 14 (such as a word processor application) at 14, headed by a security header 26.

The application 24 will be stored in encrypted form, as is conventional, and may also be compressed. Consequently, the application 24 cannot be used until it has been converted to an executable form. Conventionally, this would be controlled by the security header 26, which would cause the application 24 to be decrypted and decompressed after making various security checks, such as to check software licence details held within the computer system 10, details of the user, the current date and time (in the event the software is licensed for a specified period), and the like. Once those checks had been successfully completed, the header 26 would decrypt and decompress the application 24 and move the program pointer of the processor 12 to the beginning of the application 24, thereby handing control of the system 10 to the application 24. This type of protection can be effective in many circumstances, but is vulnerable to a hacker circumventing one or more of the checks made by the security header 26 by inserting JUMP commands, as described above, thereby enabling the application 24 to be run without all of the normal security checks having been made.

The security block 22 is provided to address this deficiency.

The security block 22 is a block of code which provides two principal functions. First, one or more security checks of the type normally provided by a security header 26 are made when the block 22 is executed. These checks may duplicate checks made by the header 26, or be in addition to them. The manner in which security checks are divided between the security header 26 and the block 22 can be varied widely and is not itself an important aspect of the invention.

However, in accordance with the invention, the function of decrypting

the application 24 is no longer instructed by the header 26, but is under the control of the block 22. Decryption of the application 24 is preferably the final act of the block 22 when executed, following which control is handed from the block 22 to the decrypted application 24.

In this way, decryption of the application 24 cannot occur unless the block 22 successfully reaches the end of its procedures, allowing decryption to be authorised and implemented.

The security block 22 is therefore a likely target for a hacker seeking to circumvent the protection provided by the invention. A successful attack on the block 22 is prevented as follows.

The security block 22 is initially (prior to execution) held in encrypted form. Encryption is by means of an encryption algorithm which requires at least one conversion key for decryption. Many encryption algorithms are known which require conversion keys for decryption. Conversion keys are commonly numerical or are used in a numerical fashion within the algorithm after conversion to numerical form, such as from machine code values of alpha-numeric characters forming the conversion key. One class of algorithms which can be used with the invention are known as CRC (cyclic redundancy check) algorithms. A CRC algorithm assesses a target block of code to produce a numerical value which is characteristic of the target block. Thus, a simple CRC algorithm might add together the machine code values of each character of the target block of code. Other CRC algorithms could be much more sophisticated than a simple addition process. However, it is important to note that each time the CRC algorithm is applied to the target block of code, the same value will be returned, whereas any change within the target block will cause a different value to be returned.

In this example, the security block 22 is held in encrypted form by use of a CRC encryption algorithm. A decryption step is thus required when the block 22 first runs. This decryption seeks to decrypt the remainder of the block 22

from a non-executable to an executable form. This is achieved by executing a CRC decryption routine based on a conversion key which is derived as a CRC value from a target block. The target block is preferably the encrypted block 22.

The consequences of these arrangements can be best understood by explaining the sequence of events when the application 24 is called.

Initially, the program pointer of the processor 12 will be pointed at the beginning of the security header 26, which will execute security checks before handing control to the block 22. At this stage, the application 24 is still in encrypted form. The block 22 is also in an encrypted form and is first decrypted. Decryption is achieved by an algorithm which requires a conversion key. The conversion key is the CRC value derived from the block 22 itself. If the block 22 has not been modified, an appropriate CRC value will be returned to allow conversion. The block 22 can then run normally, as a result of which, the application 24 will be decrypted by operation of the block 22. After the block 22 has fully executed, control will be handed to the application 24, which then runs in the normal manner. However, in the event that the block 22 has been modified in any way, the CRC value of the block 22 will have changed. As a result, the value used as a conversion key in the decryption algorithm will not be appropriate for correct decryption. The decrypted version of the block 22 will be corrupt. In any real situation, it is expected that any change to the conversion key would cause sufficient corruption as to prevent any further execution of the block 22. This would leave the application 24 in encrypted form.

Any attempt to use a debugger program to analyse the block 22 will cause the introduction of interrupt commands within the block 22, as has been described above. The act of using the debugger to look at the block 22 will therefore itself change the CRC value of the block 22, even before the debugger is used to make changes. The change to the CRC value will prevent the application 24 from being decrypted. Thus, operation of the application 24 is

prevented by an attempt to look at the security provided by the block 22, even if no modification of the security block is made as a result of the inspection.

A hacker seeking to circumvent security will be most interested in looking at those blocks of code which appear to be responsible for security checks. It is therefore preferred that the target block used for producing a CRC value for decryption of the security block 22 includes at least part of the block which conducts one or more security checks prior to authorising decryption of the application 24. Thus, all of these security checks could be executed by the decrypted block 22, in which case, the CRC conversion key could be derived from all or part of the encrypted block 22. Alternatively, some checks could be executed by the block 22, in which case the CRC value could be derived wholly or partly from all or part of the header 26.

Example 2

Fig. 3 illustrates a more complex alternative implementation of the invention. Many features illustrated in Fig. 3 correspond with the features of Fig. 2, and like numerals are used for these features.

The principal difference between the implementations of Figs. 2 and 3 is the manner of encryption of the security block 22. The block 22 is illustrated as a plurality of sub-blocks 30. The sub-blocks 30 together provide an encrypted form of the block 22, but each sub-block 30 is separately encrypted. Each may be encrypted by the same algorithm and a respective conversion key, or by different algorithms, each requiring the same conversion key, or by different algorithms requiring different conversion keys.

When the example of Fig. 3 is run, the sequence of operations is largely the same as that described above in relation to Fig. 2, except as follows. When the block 22 is first called, a decryption algorithm is executed to decrypt only the uppermost sub-block 30. This may be by means of a CRC value of the sub-block being decrypted, or another sub-block, or a target block elsewhere (such

as all or part of the security header 26). After decryption, the first sub-block 30 can then be executed. This may implement security checks, or other initial routines. Execution of the first sub-block 30 concludes with an instruction to decrypt the second sub-block 30. This is achieved by a decryption algorithm requiring a conversion key which will preferably be derived at the time of decryption, even if the conversion key is the same as the conversion key used for decrypting the first sub-block 30. Thus, any interference in the target block since decryption of the first sub-block will cause a change in the CRC code, and thus cause corrupt decryption of the second sub-block 30.

These operations are illustrated more fully in Fig. 4. Fig. 4 illustrates three lines of code at the boundary between the first and second sub-blocks 30A, 30B. Sub-block 30A concludes at line 99 with a final line, the nature of which is not important to the invention. Line 100 is then executed. This is a decryption instruction. The instruction specifies the block of data to be decrypted, here lines 101 to 200, which represent the second sub-block 30B. Instruction 100 also identifies the algorithm to be used, here algorithm A. Finally, instruction 100 identifies the conversion key to be used by specifying the lines (here specified as lines 101 to 200) to be used as a target block to derive a conversion key for use within algorithm A.

Once line 100 has been executed, execution will move to line 101. In the event that line 100 achieved successful decryption, line 101 will correctly indicate the first action of the second sub-block 30B. Successful decryption would occur if the sub-block 30B had not been modified, so that the correct CRC value was used as a decryption key. However, if the second sub-block 30B had been modified, or was being inspected by a debugger program at the time the CRC value was being calculated, line 101 the wrong CRC value would be used for decryption, the sub-block 30B would be corrupt, and execution would cease at that point. This would leave the application 24 in encrypted form.

As execution of the security block 22 continues, each sub-block 30 is decrypted in its turn by the preceding sub-block, in the manner just described

and as indicated by the arrows 32 in Fig. 3. In the event that there is no corruption in the block 22, and a debugger program is not being used, the block 22 will fully decrypt in uncorrupted form, and execute successfully, reaching the end of the final sub-block 30, illustrated in Fig. 5. At the end of the final sub-block 30, that is, at the end of execution of the block 22, there is an instruction (line 998) to decrypt the application 24, followed by an instruction (line 999) to go to the application 24. The decryption at 998 can be based on an algorithm requiring a decryption key derived as a CRC value as described above, in order to further apply the principles of the invention. Line 999 will only be reached and executed correctly if every sub-block 30 has been correctly decrypted.

Example 3

The two examples described above assume that execution of the software commences with the security block 22 separately installed in the RAM 16, as illustrated in Figs. 2 and 3. The third example, illustrated in Figs. 6A and 6B includes a further decryption and decompression step, as follows.

Initially, the RAM 16 includes a single block of software 20A (Fig. 6A). This incorporates the encrypted application 24, the security header 26, and a security stub 40, illustrated in this example as attached to the end of the application 24. The security stub 40 incorporates a compressed and encrypted version of the security block 22 of Examples 1 or 2. The security header 26 includes an instruction to decrypt and/or decompress the stub 40 and to install the result as a separate block, forming the security block 22 (Fig. 6B). When the security header 26 has fully executed, control will then pass to the security block 22. Subsequent execution will proceed as described above in relation to Example 1 or Example 2.

An advantage of this example is to allow the stub 40 to be provided as part of the software licensed to a legitimate user, so that the block 20 as illustrated in Fig. 3A will be installed, including the stub 40, when the licensed

application is installed. This ensures that a properly licensed version of the application 24 is always accompanied by the security stub 40, thus ensuring that the authorised copy remains protected by the benefits of the software protection arrangements of the invention.

It will be apparent that many variations and modifications can be made to the embodiments described above, without departing from the scope of the present invention. In particular, many different algorithms for decryption and for calculation of CRC values could be used.

Whilst endeavouring in the foregoing specification to draw attention to those features of the invention believed to be of particular importance it should be understood that the Applicant claims protection in respect of any patentable feature or combination of features hereinbefore referred to and/or shown in the drawings whether or not particular emphasis has been placed thereon.

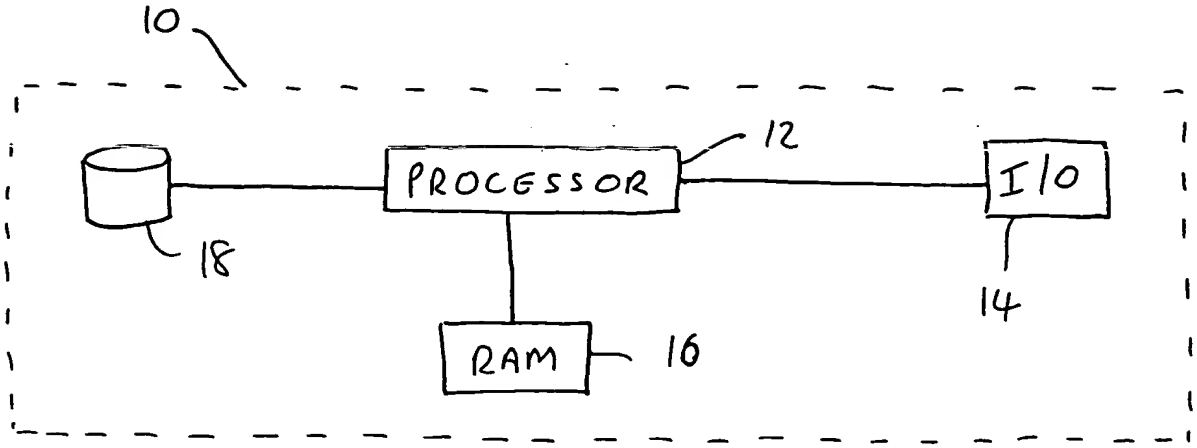

FIG. 1

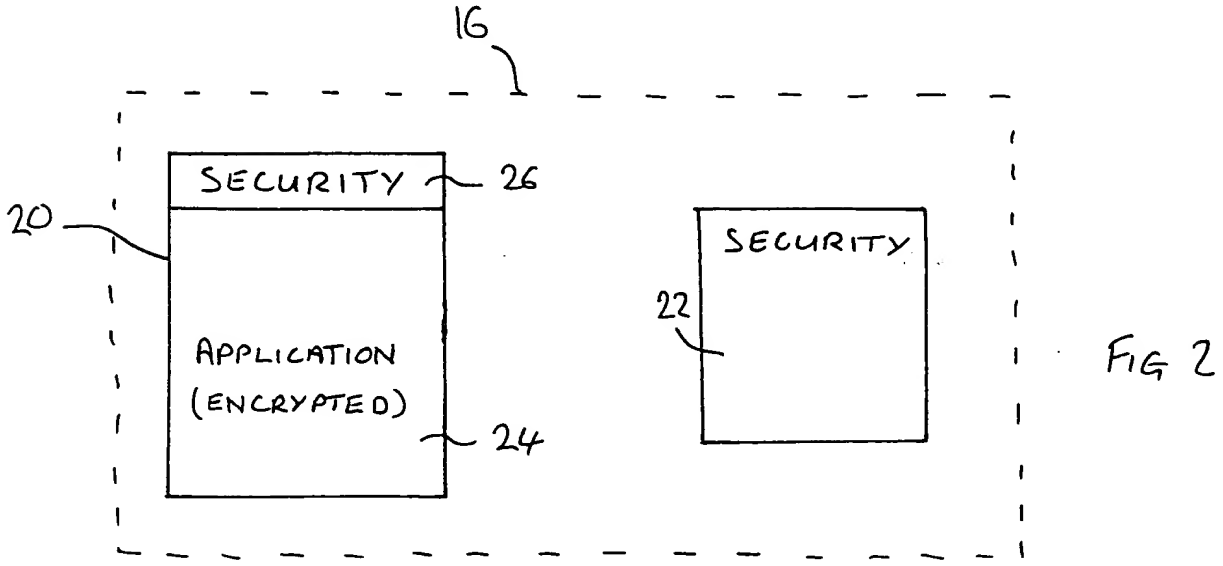

FIG 2

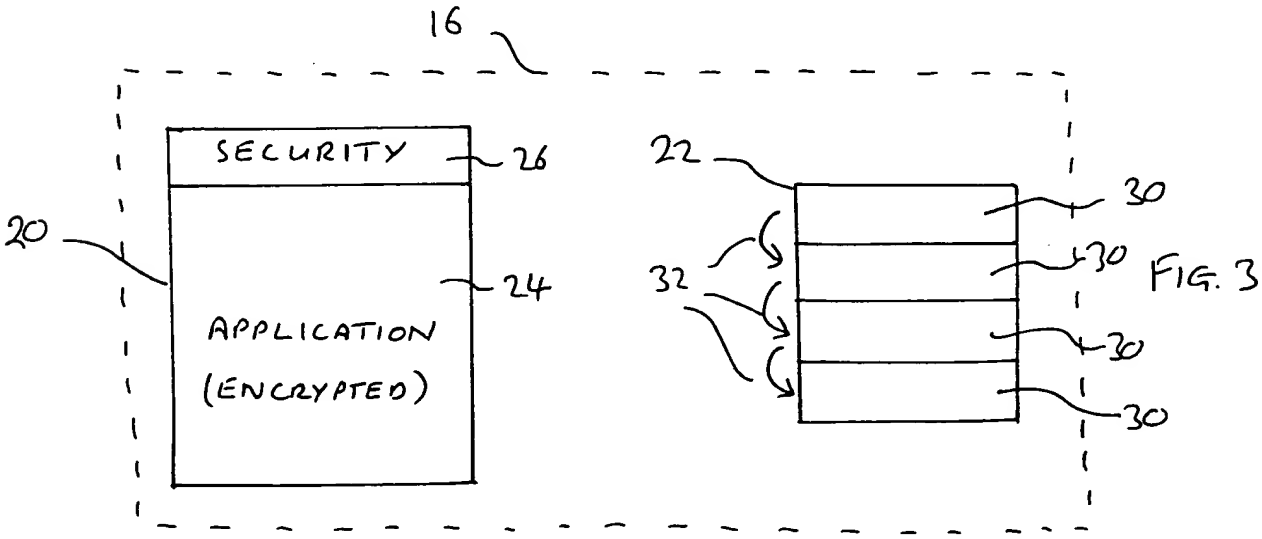

FIG. 3

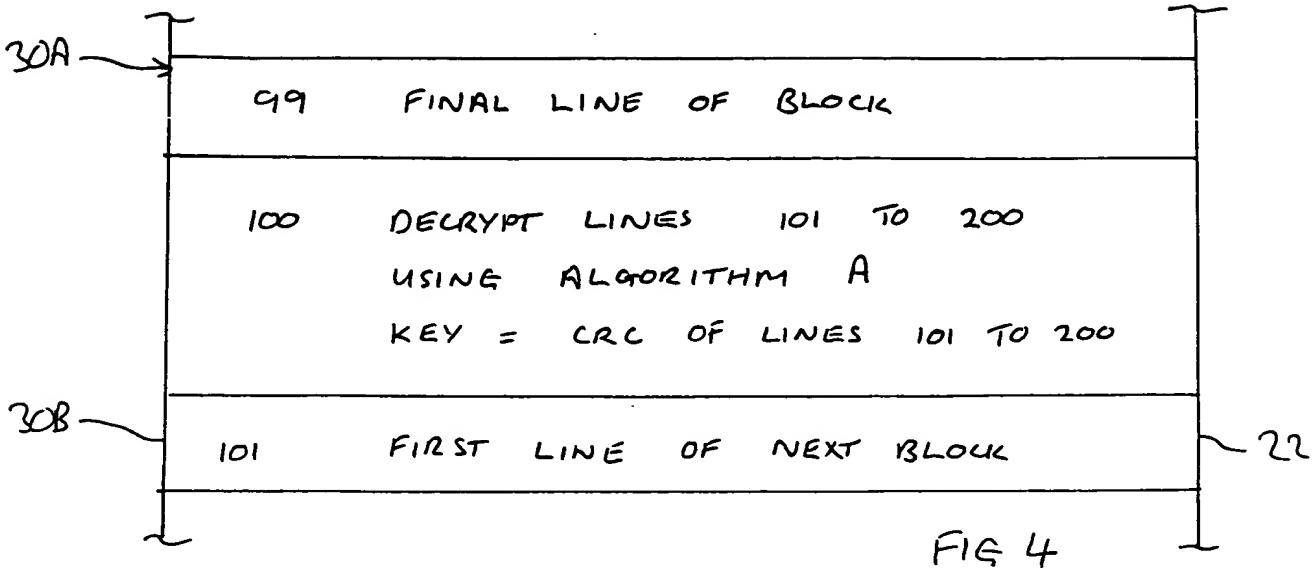
30A
99 FINAL LINE OF BLOCK
100 DECRYPT LINES 101 TO 200
USING ALGORITHM A
KEY = CRC OF LINES 101 TO 200
30B
101 FIRST LINE OF NEXT BLOCK
22

FIG 4

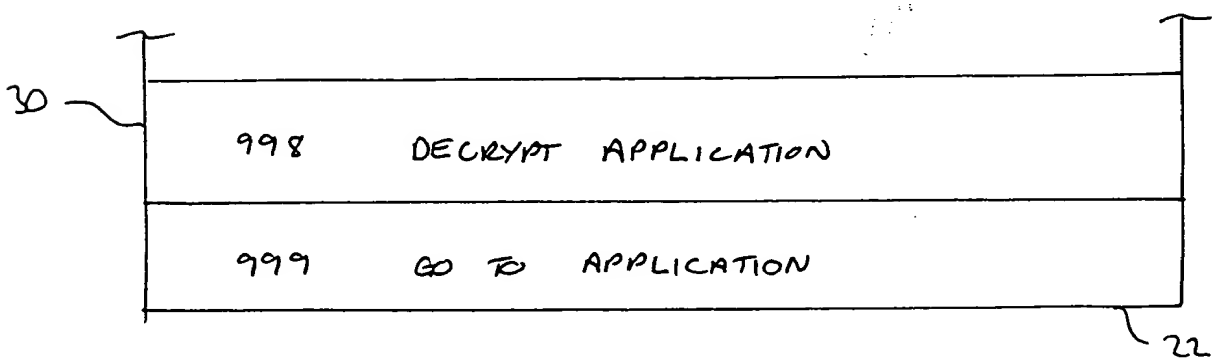
30
998 DECRYPT APPLICATION
999 GO TO APPLICATION
22

FIG 5

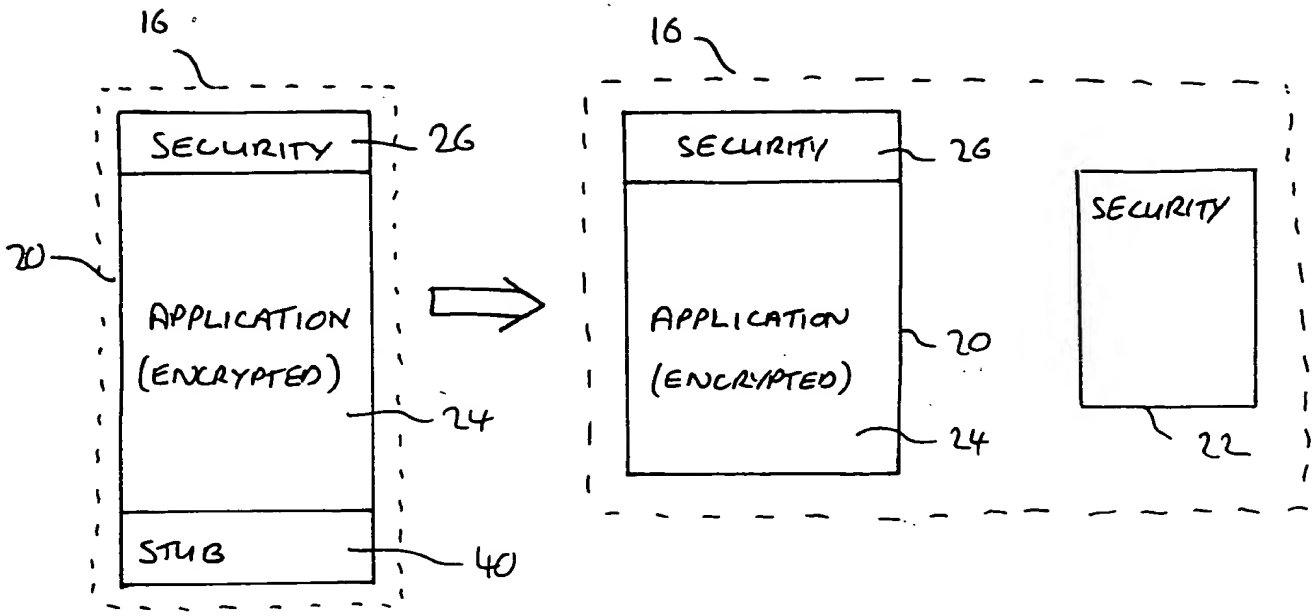
16
SECURITY
26
20
APPLICATION
(ENCRYPTED)
24
STUB
40
16
SECURITY
26
APPLICATION
(ENCRYPTED)
20
24
SECURITY
22

FIG 6A

FIG 6B